View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by NASA Technical Reports Server

Numerical Evaluation of the "Dual-Kernel Counter-flow" Matric Convolution Integral that Arises in Discrete/Continuous (D/C) Control Theory

Douglas D. Nixon
NASA, Marshall Space Flight Center
Huntsville, Alabama 35812 USA

Abstract-**Discrete/Continuous (D/C) control theory is a new generalized theory of discrete-time control that expands the concept of conventional (exact) discrete-time control to create a framework for design and implementation of discrete-time control systems that include a continuous-time command function generator so that actuator commands need not be constant between control decisions, but can be more generally defined and implemented as functions that vary with time across sample period. Because the plant/control system construct contains two linear subsystems arranged in tandem, a novel "dual-kernel counter-flow" convolution integral appears in the formulation. As part of the D/C system design and implementation process, numerical evaluation of that integral over the sample period is required. Three fundamentally different evaluation methods and associated algorithms are derived for the constant-coefficient case. Numerical results are matched against three available examples that have closed-form solutions.**

I. Introduction

As introduced in [1], Discrete/Continuous (D/C) control theory defines a new generalized approach to design of discrete-time control systems that, unlike traditional discrete-time control, does not restrict actuator commands to be constant over the control decision interval , but allows them to vary continuously with time, in a "smart" manner, across each sample period between decisions. Because the system construct contains two linear subsystems that operate in tandem, a novel dual-kernel counter-flow matric convolution integral naturally arises in the formulation that, in its most general form, is a matrix defined by

$$\overline{D}(t_2,t_1)=\int_{t_1}^{t_2}[\Phi_p(t_2,\tau)B_p(\tau)C_c(\tau)\Phi_c(\tau,t_1)]d\tau\,, \quad (1)$$

where (t_2,t_1) are times associated with extremes of the sample period, (Φ_p,Φ_c) are state transition matrices associated with the plant and control system, $B_p(t)$ is the plant input distribution matrix, $C_c(t)$ is the control system output selection matrix, and the definition and form have been taken from [1].

Specializing (1) to the constant coefficient case yields

$$\overline{D}(T)=\int_0^T e^{A_p(T-\tau)}B_pC_ce^{A_c(\tau)}d\tau\,, \quad (2)$$

where the constants A and B are associated with the standard state-variables representation of linear system dynamics as indicated by

$$\dot{x}=Ax+Bu \quad (3)$$

and

$$y=Cx\,, \quad (4)$$

the subscripts p and c, as before, refer to "plant" and "control", and T is the time between control decisions (sample period). A block diagram representation of the system theoretical construct is shown in Fig. 1.

Design and computation of the control gain-matrix $\tilde{K}$ shown in Fig. 1 requires evaluation of the convolution integral $\overline{D}$ [1, 2]. The rest of this paper is devoted to development of three specific approaches to numerical evaluation of $\overline{D}$. The first method relies on series expansion and truncation for the individual plant and control system state transition matrices, and also makes use of the state transition matrices to partition the sample period so that the length of time for which the series expansions must be valid can be less than the sample period. A second method, which can be described as a

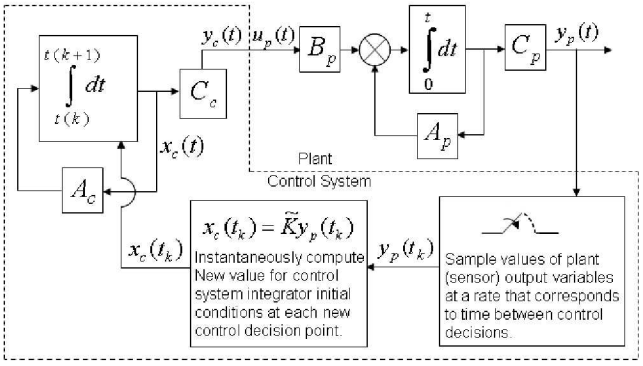

Fig. 1. Diagram of Plant/Control System

mapping-matrix approach, requires simulation of the system to generate plant time responses across the sample period as a function of control system initial conditions. The simulation is run as many times as there are control system states. The third method relies on a single series expansion and truncation for the total system state transition matrix followed by extraction of the convolution as a partitioned block of the matrix result. While this method involves indirect computation of $\overline{D}$ from the solution of the total system matric differential equation, Johnson [3] describes a potential fourth approach that involves identification of a matric differential equation whose solution directly produces $\overline{D}$.

II. Development of Methods and Algorithms

A. Method 1: Sample-Period-Partitioned Double Series Expansion

The first method and algorithm are derived by working directly with the definition of $\overline{D}$ given in (2). Both exponential functions are replaced with their infinite series counterparts so that

$$\overline{D} = e^{A_p T} \int_0^T \left[\sum_{n=0}^{\infty} \frac{(-A_p \tau)^n}{n!} \right] B_p C_c \left[\sum_{m=0}^{\infty} \frac{A_c{}^m \tau^m}{m!} \right] d\tau. \quad (5)$$

Exchanging integration with summation, a single element of the resulting (double) summation is expressed by,

$$\overline{d}_{mn} = e^{A_p T} \frac{A_p^n B_p C_c A_c^m (-1)^n}{m!\,n!} \int_0^T \tau^{m+n} d\tau, \quad (6)$$

where $\overline{d}_{mn}$ is any element of $\overline{D}$. Now the integration can be performed analytically, and the corresponding general element of $\overline{D}$ is given by

$$\overline{d}_{mn} = e^{A_p T} \frac{(-1)^n A_p^n B_p C_c A_p^m}{m!\,n!\,(m+n+1)} \tau^{m+n+1} \Big|_0^T. \quad (7)$$

Finally, the integral $\overline{D}$ can be expressed as

$$\overline{D}(T) = e^{A_p T} \sum_{m=0}^{M} \sum_{n=0}^{N} \frac{(-1)^n A_p^n B_p C_c A_c^m T^{m+n+1}}{m!\,n!\,(m+n+1)}, \quad (8)$$

where the integers N and M represent the arbitrarily large but finite number of terms included in the series plant and Φ_c of the control system. expansions for the state transition matrices Φ_P of the

Equation (8) potentially represents an algebraic definition of requirements for the detailed design of a computer algorithm, which can be implemented in a language such as Fortran, C, or Matlab®, to numerically evaluate $\overline{D}$. However, for reasons of practical experience, the algorithm derivation will be taken through another major step. The time between control decisions T will be partitioned into an arbitrary number of sub-intervals q, and the convolution integral $\overline{D}(T)$ will be expressed as a function of $\overline{D}(T/q)$ and the state transition matrices $\Phi_P(T/q)$ and $\Phi_c(T/q)$. Because the final symbolic expression for $\overline{D}(T)$ will explicitly include both the number of subintervals q and the number of terms N and M used in the exponential series expansions, the algorithm will become more tunable for specific circumstances and much more capable of handling long sample periods. Choosing values for the tuning parameters N, M, and q is loosely analogous to choosing order in time and length of time-step associated with numerical integration of differential equations. This form of the algorithm will be referred to as sample-period-partitioned (SPP).

To facilitate derivation of the SPP algorithm, a standard compact discrete-time control notation is employed and explained below. For constant input u over the sample period T, often referred to as sample-and-hold or zero-order-hold (ZOH), the exact discrete-time equivalent to the continuous-time system represented by (3) is

$$x(k+1) = \widetilde{A}(T)x(k) + \widetilde{B}(T)u(k) \quad (9)$$

for $k = 0, 1, 2, \cdots$, where $\widetilde{A}$ is the state-transition matrix

$$\widetilde{A}(T) = e^{AT}, \quad (10)$$

and $\widetilde{B}$ is the discrete input distribution matrix

$$\widetilde{B}(T) = \int_0^T e^{A(T-\tau)} B d\tau. \quad (11)$$

The tacit assumption of constant sample period, $T(k) = T$, in (9-11) is a notational convenience that affords no significant loss of generality.

As developed in [1], the D/C generalization of (9) for the plant is

$$x_p(k+1) = \widetilde{A}_p(T)x_p(k) + \overline{D}(T)x_c(k), \quad (12)$$

where the value of $x_c(k)$ is updated from the D/C control computations at each sample-time $t_k = kT$ based on $x_p(k) = x_p(kT)$ and the D/C control gains associated with a particular design. The continuous variation of the D/C control system state as it is propagated across the sample period is given by

$$x_c(t) = e^{A_C(t-kT)} x_c(kT); \quad kT \le t \le (k+1)T. \quad (13)$$

In the more compact notation of (12), an evaluation of the control state at the end of the sample period, prior to any potential alteration (update) by the D/C control logic, is given by

$$x_c(k+1) = \tilde{A}_c(T)x_c(k). \tag{14}$$

Equations (12) and (14) form a basis from which the sample period can be partitioned. To this end, an integer q is defined such that it represents the number of sample period sub-intervals resulting from the partitioning. Then (12) applied to the first sub-interval becomes

$$x_p(k+1/q) = \tilde{A}_p(T/q)x_p(k) + \overline{D}(T/q)x_c(k). \tag{15}$$

Similarly, (14) becomes

$$x_c(k+1/q) = \tilde{A}_c(T/q)x_c(k). \tag{16}$$

Continuing forward in time,

$$x_p(k+2/q) = \tilde{A}_p(T/q)x_p(k+1/q) + \overline{D}(T/q)x_c(k+1/q). \tag{17}$$

Substituting (15) and (16) into (17) and rearranging,

$$x_p(k+2/q) = \tilde{A}_p^2(T/q)x_p(k) + [\tilde{A}_p(T/q)\overline{D}(T/q) + \overline{D}(T/q)\tilde{A}_c(T/q)]x_c(k), \tag{18}$$

where, in the interest of clarity, it is noted that

$$\tilde{A}(T) = \tilde{A}^q(T/q) = [e^{AT/q}]^q, \tag{19}$$

for both the plant and control systems. Continuing a similar algebraic process,

$$x_p(k+3/q) = \tilde{A}_p^3(T/q)x_p(k) + [\tilde{A}_p^2(T/q)\overline{D}(T/q) + \tilde{A}_p(T/q)\overline{D}(T/q)\tilde{A}_c(T/q) + \overline{D}(T/q)\tilde{A}_c^2(T/q)]x_c(k), \tag{20}$$

and

$$x_p(k+4/q) = \tilde{A}_p^4(T/q)x_p(k) + [\tilde{A}_p^3(T/q)\overline{D}(T/q) + \tilde{A}_p^2(T/q)\overline{D}(T/q)\tilde{A}_c(T/q) + \tilde{A}_p(T/q)\overline{D}(T/q)\tilde{A}_c^2(T/q) + \overline{D}(T/q)\tilde{A}_c^3(T/q)]x_c(k). \tag{21}$$

Then, by induction

$$x_p(k+1) = \tilde{A}_p^q(T/q)x_p(k) + [\sum_{n=1}^{q} \tilde{A}_p^{n-1}(T/q)]\overline{D}(T/q)\tilde{A}_c^{q-n}(T/q)]x_c(k). \tag{22}$$

Finally, by comparison with (12),

$$\overline{D}(T) = \sum_{n=1}^{q} \tilde{A}_p^{n-1}(T/q)]\overline{D}(T/q)\tilde{A}_c^{q-n}(T/q), \tag{23}$$

where, from (8)

$$\overline{D}(T/q) = e^{Ap(T/q)} \sum_{m=0}^{M}\sum_{n=0}^{N} \frac{(-1)^n A_p^n B_p C_c A_c^m (T/q)^{m+n+1}}{m!n!(m+n+1)}, \tag{24}$$

and it should be pointed out that the notation (T/q) on the right-hand side of (24) simply indicates an algebraic parameter grouping, and is different from but consistent with the functional notation (T/q) used previously and on the left-hand side of (24) . The complete sample-period partitioned algorithm, defined by (23) and (24), can now be applied over a very long control system decision-making interval (large T) because the computational process is explicitly tunable in terms of both the order M and N of the series expansions and the associated time interval T/q .

B. Method 2: Mapping-Matrix Construction by Continuous System Simulation

The role of $\overline{D}$ in a discrete-continuous control system is to map the contribution of current states of the control system across the control decision interval (sample period) to future states of the plant. Thus, responses of a continuous model of the plant, driven by output from the (continuous) function generator portion of the D/C control system, can be repetitively propagated across the sample period, by dynamic simulation, to generate data from which $\overline{D}$ can be constructed. A block diagram of the simulation construct is shown in Fig. 2.

For an n-dimensional plant and an m-dimensional control system, the dimension of $\overline{D}$ is $n \times m$, and m simulation runs are required to construct $\overline{D}$. Because absolute scale is of no theoretical concern for a linear system, the columns of an $m \times m$ identity matrix can be used to define an appropriate m-dimensional initial state vector for the function generator for each of the m simulation runs. Initial values of the plant states are set to zero so that the contribution to plant response from the

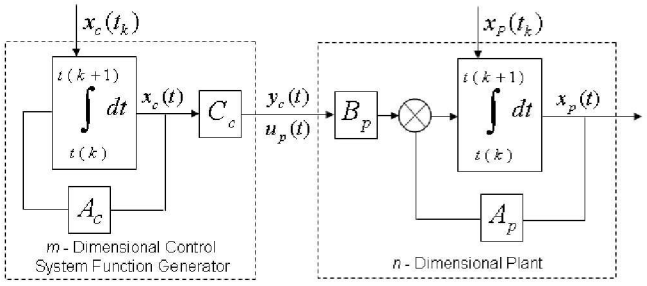

Fig. 2. Diagram of Continuous Simulation

function generator alone is evaluated. Each simulation run produces final values for n elements of the plant state vector, and $\overline{D}$ can be constructed from the resulting data set. The basic mathematical relationship supporting the computational process is

$$x_p(T) = \overline{D}(T)x_c(0). \tag{25}$$

Then,

$$\overline{D}_{(nxm)}(T) = [x_p^1(T) \,\vdots\, x_p^2(T) \,\vdots\, \cdots \,\vdots\, x_p^m(T)]_{(nxm)}. \tag{26}$$

Because the initial control system states are the columns of an m-dimensional identity matrix, the convolution integral is given by

$$[x_p^1(T) \,\vdots\, x_p^2(T) \,\vdots\, \cdots \,\vdots\, x_p^m(T)]_{(nxm)} = \overline{D}_{(nxm)}(T)\,[x_c^1(0) \,\vdots\, x_c^2(0) \,\vdots\, \cdots \,\vdots\, x_c^m(0)]_{(mxm)}. \tag{27}$$

where the $x_P^i(T)$ are the n-dimensional plant state vectors at time $t = T$ which have been propagated by the i^{th} execution of the continuous system dynamic simulation of Fig. 2 with initial conditions $x_P^i(0)$ and $x_c^i(0)$ given by

$$[x_p^1(0) \,\vdots\, x_p^2(0) \,\vdots\, \cdots \,\vdots\, x_p^m(0)]_{(nxm)} = 0, \tag{28}$$

and

$$[x_c^1(0) \,\vdots\, x_c^2(0) \,\vdots\, \cdots \,\vdots\, x_c^m(0)]_{(mxm)} = I_{(mxm)}. \tag{29}$$

C. Method 3: Sample-Period-Partitioned Single Series Expansion

A combined state-variable representation of the plant and D/C control system function generator depicted in Fig. 2 can be represented by

$$\dot{X} = A_{sys}X, \tag{30}$$

where

$$X = \begin{bmatrix} x_p \\ \hdashline x_c \end{bmatrix}, \tag{31}$$

and

$$A_{sys} = \begin{bmatrix} A_p & \vdots & B_pC_c \\ \hdashline 0 & \vdots & A_c \end{bmatrix}. \tag{32}$$

Then, the exact discrete-time representation is

$$X((k+1)T) = \tilde{A}_{sys}X(kT), \tag{33}$$

where

$$\tilde{A}_{sys} = e^{A_{sys}T}. \tag{34}$$

Then, by inspection, comparisons, and substitutions among (12) and (30-34), it becomes clear that

$$\tilde{A}_{sys} = \begin{bmatrix} \tilde{A}_p & \vdots & \overline{D} \\ \hdashline 0 & \vdots & \tilde{A}_c \end{bmatrix}, \tag{35}$$

where $\overline{D}$ is as defined by (2). Then, an obvious method for computing $\overline{D}$ is to compute $\tilde{A}$ using any of several known standard methods [4], and extract $\overline{D}$ from the partitioned result.

IV. Implementation

Algorithm functionality for Method 1 is defined by (23) and (24). Using this definition as a guide, the process was programmed directly using the core capability of Matlab®.

For Method 2, a combination of Matlab® and Simulink® was used to build the simulation described in Fig. 2, manage its execution, store results, and perform subsequent computations as indicated by (27)–(29).

Algorithm functionality for Method 3 is defined by (32), (34), and (35), and was implemented in Matlab®. However, two different mathematical processes were used to compute the state transition matrix $\tilde{A}$ defined in (35). The first is an SPP series-expansion approach derived from truncating a series expansion of $\tilde{A}$ analogous to that shown in (19), while the second uses the Matlab® library function *expm*( ).

V. Functional Verification

Initial verification was accomplished by using all three methods to evaluate $\overline{D}$ for three specific cases whose closed-form solutions are available [5] and [6]. With reasonably proper selection of algorithm parameters, all methods appear to function correctly. Algorithm parameters include number of terms retained in series expansions, number of sample-period partitions, and maximum allowable integration step size in the case of simulation. The closed-form test cases are outlined below.

A. Case I: Harmonic Oscillator Plant with Constant-Only D/C Control Basis Functions (ZOH control).

This system consists of an un-damped second-order oscillatory plant and a control system whose function generator is only capable of producing actuator commands that are constant across the sample period. This case is equivalent to the ZOH (Zero Order Hold) associated with traditional discrete-time control. The matrices that define this example system are given in [5] as

$$\{A_P, B_P\} = \left\{ \begin{bmatrix} 0 & 1 \\ -\omega_{np}^2 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \tag{36}$$

and

$$\{A_c, C_c\} = \{0, 1\}, \tag{37}$$

where ω_{np} is the plant natural frequency. The closed-form (reference) expression for $\overline{D}$ in this case is [5]

$$\overline{D}_{ref} = \widetilde{B}_p = \begin{bmatrix} \omega_{np}^{-2}[1 - \cos(\omega_{np} T)] \\ \omega_{np}^{-1} \sin(\omega_{np} T) \end{bmatrix}. \tag{38}$$

Relative computational accuracy can be measured as an error norm defined by

$$error = \frac{\left\| \overline{D} - \overline{D}_{ref} \right\|}{\left\| \overline{D}_{ref} \right\|}, \tag{39}$$

where $\| ( ) \|$ can be any suitable matrix norm, and was evaluated using the "largest singular value" (default) option associated with the Matlab® function *norm*( ). Algorithm error results, along with pertinent parameter values, are shown in Table I.

B. Case II: Harmonic Oscillator Plant with Linear-in- Time (LiT) Control Basis Functions

This system, like Case I, consists of an un-damped second-order oscillatory plant as described by (35), but the control system function generator is capable of producing LiT actuator commands (constant plus ramp) across the sample period. The function generator in this example is defined by [6]

$$\{A_c, C_c\} = \left\{ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \end{bmatrix} \right\}, \tag{40}$$

and the closed-form expression for $\overline{D}$ from [6] is

$$\overline{D} = \begin{bmatrix} \omega_{np}^{-2}[1 - \cos(\omega_{np} T)] & \omega_{np}^{-3}[\omega_{np} T - \sin(\omega_{np} T)] \\ \omega_{np}^{-1} \sin(\omega_{np} T) & \omega_{np}^{-2}[1 - \cos(\omega_{np} T)] \end{bmatrix}. \tag{41}$$

Computational accuracy is measured according to (39); algorithm parameters and error results are shown in Table II.

C. Case III: Double Integrator Plant with Three-State Control function Generator.

This system consists of a double integrator plant, defined by

$$\{A_p, B_p, \} = \left\{ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}, \tag{42}$$

and a control function generator defined by

$$\{A_c, C_c\} = \left\{ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -\omega_{nc}^2 & 0 \end{bmatrix}, [1 \quad 0 \quad 0] \right\}, \tag{43}$$

where ω_{nc} is the frequency associated with the control system basis functions. The closed-form expression for $\overline{D}$ for this system is [6]

$$\overline{D} = \begin{bmatrix} T^2/2 & \omega_{nc}^{-3}[\omega_{nc} T - \sin(\omega_{nc} T)] \\ T & -\omega_{nc}^{-2}[\cos(\omega_{nc} T) - 1] \\ & \omega_{nc}^{-2} T^2/2 + \omega_{nc}^{-4}[\cos(\omega_{nc} T) - 1] \\ & \omega_{nc}^{-3}[\omega_{nc} T - \sin(\omega_{nc} T)] \end{bmatrix}, \tag{44}$$

Computational accuracy is measured as before, and algorithm parameters and error results are shown in Table III.

TABLE I
FUNCTIONAL TEST RESULTS, HARMONIC OSCILLATOR, ZOH CONTROL

Parameter Set #		1	2	3	4
Plant Parameters		$\omega_{np} = 1$	$\omega_{np} = 2$	$\omega_{np} = 3$	$\omega_{np} = 4$
Control System Parameters		$T = 0.01$	$T = 0.1$	$T = 1$	$T = 10$
Algorithm Parameters*		$N = M = 10$ $q = 1$	$N = M = 4$ $q = 10$	$N = M = 10$ $q = 100$	$N = M = 10$ $q = 1000$
Error Norm	Double Series	1.4427e-015	2.6623e-016	2.3457e-015	3.6833e-015
	Simulation	1.4878e-015	7.8487e-017	2.3969e-013	2.3633e-012
	Single Series, SPP	1.4413e-015	2.6512e-015	1.6612e-015	2.3159e-014
	Single Series, *expm( )*	1.4544e-015	1.4062e-016	7.7714e-016	3.2019e-015

*M applies only to dbl. series method; max. stepsize=5e-04 for all simulations.

TABLE II
FUNCTIONAL TEST RESULTS, HARMONIC OSCILLATOR, LiT CONTROL

Parameter Set #		1	2	3	4
Plant Parameters		$\omega_{np} = 1$	$\omega_{np} = 2$	$\omega_{np} = 3$	$\omega_{np} = 4$
Control System Parameters		$T = 0.01$	$T = 0.1$	$T = 1$	$T = 10$
Algorithm Parameters*		$N = M = 6$ $q = 1$	$N = M = 7$ $q = 10$	$N = M = 10$ $q = 100$	$N = M = 10$ $q = 1000$
Error Norm	Double Series	1.4631e-015	1.3963e-016	1.7847e-015	4.8308e-015
	Simulation	1.6502e-015	8.3737e-017	1.8350e-013	7.9146e-013
	Single Series, SPP	1.4899e-015	1.3936e-016	1.2828e-015	7.7684e-015
	Single Series, *expm( )*	1.5485e-015	1.4172e-016	5.9622e-016	1.4203e-015

*M applies only to dbl. series method; max. stepsize=5e-04 for all simulations.

TABLE III
FUNCTIONAL TEST RESULTS, DOUBLE INTRGRATOR, 3-STATE CONTROL

Parameter Set #		1	2	3	4
Plant Parameters		$\omega_{nc}=1$	$\omega_{nc}=2$	$\omega_{nc}=3$	$\omega_{nc}=4$
Control System Parameters		$T=0.01$	$T=0.1$	$T=1$	$T=10$
Algorithm Parameters*		$N=M=6$ $q=1$	$N=M=7$ $q=10$	$N=M=10$ $q=100$	$N=M=10$ $q=1500$
Error Norm	Double Series	1.4425e-015	1.3953e-016	5.9627e-016	1.8218e-015
	Simulation	1.6051e-015	9.8446e-017	3.7764e-014	2.0503e-013
	Single Series, SPP	1.4617e-015	1.3939e-016	6.5836e-016	8.6898e-015
	Single Series, *expm()*	1.4530e-015	1.8313e-017	2.9141e-017	3.6425e-017

*M applies only to dbl. series method; max. stepsize=5e-04 for all simulations.

uses Matlab® library function *exp*() to compute the matrix exponential seems to produce the most accurate and efficient process most of the time. As might be expected, the simulation method is generally much less efficient than the other methods, and its accuracy noticeably diminishes as sample period increases. The double series expansion method, while less efficient than the single series method, appears to have significant potential for improvement at the code level. It also provides a symbolic formula for direct evaluation of the convolution in terms of plant, control system, and algorithm tuning parameters. It is possible that this feature could be of further use in D/C control development. For the constant-coefficient and constant-sample-period case, the computation is needed only once, and can be done before-the-fact, so efficiency may be of no real concern.

VI. SUMMARY AND CONCLUSIONS

Three fundamentally different methods for numerical evaluation of the dual-kernal counter-flow convolution integral associated with Discrete/ Continuous control theory have been derived and implemented. The first method relies on series expansion and truncation for the individual plant and control system state transition matrices, and also makes use of the state transition matrices to partition the sample period so that the length of time for which the series expansions must be valid can be much less than the sample period. The second method requires simulation of the system to generate plant time responses across the sample period as a function of control system initial conditions. The simulation is run as many times as there are control system states. A third method relies on a single series expansion and truncation for the total system state transition matrix followed by extraction of the convolution as a partitioned block of the matrix result. This method also incorporates sample period partitioning. All three methods appear to accomplish the objective in a functional sense; however, they are significantly different, and are likely to have different strengths and weaknesses with respect to accuracy, efficiency and utility in future developments associated with Discrete/ Continuous control.

REFERENCES

[1] Johnson, C.D., A New Discrete-Time State Model for Linear Dynamical Systems with Continuously-Varying Control/Disturbance Inputs," *Proc. 1994 Southeastern Symposium on System Theory*, Ohio University, Athens, Ohio, March, 1994, p. 523.

[2] Johnson, C.D., "Improved Discrete-Time Disturbance-Accommodating Control Performance Using Discrete/ Continuous Control Theory; Part 1: Complete Disturbance Cancellation (Rejection)," *Proc. 2003 Southeastern Symposium on System Theory*, W. Virginia University, Morgantown, WV, pp. 127-1 33

[3] C.D. Johnson, "On the Computation of a Dual- Kernal Matric Convolution Integral", Submitted to Southeastern Symposium on System Theory, 2009

[4] C. B. Moler and C. F. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix*, SIAM Rev., 20 (1978), pp. 801–836.

[5] Johnson, C.D., "Improved Discrete-Time Controller Performance Using the 'Control-Dimension Multiplier Effect' of Discrete/ Continuous Control Theory," *Proc. of the 34th Southeastern Symposium on System Theory*, University of Alabama in Huntsville, Huntsville, AL, March 18-19, 2002.

[6] Hunt & Johnson, to be published